Walkthrough: Creating User-Selectable Themes

This walkthrough illustrates how to create an ASP.NET page that lets the user select a theme for the page. Although this example uses a single control skin and a basic cascading style sheet (CSS) file, the principles shown apply to more complex themes that include graphics, different layout schemes in the CSS file, and more complex server control skins.

Tasks illustrated in this walkthrough include:

- Creating a theme, including a CSS file and a server control skin, in Microsoft Visual Web Developer.
- Creating an ASP.NET master page that uses a theme.
- Creating an ASP.NET page with a master page applied that uses a theme.
- Creating a drop-down list server control that applies a new theme to a page, including changing styles on the master page elements.
- Running a page to show different themes applied to the page.

Prerequisites

To complete this walkthrough, you will need:

- Visual Web Developer (Visual Studio).
- The .NET Framework.

Creating a Web Site

If you have already created a Web site in Visual Web Developer (for example, by following the steps in Walkthrough: Creating a Basic Web Forms Page in Visual Studio), you can use that Web site and skip to the next section, "Creating a Master Page." Otherwise, create a new Web site and page.

This walkthrough uses a Web site project. You could use a Web application project instead. For information about the difference between these Web project types, see Web Application Projects versus Web Site Projects in Visual Studio.

To create a file system Web site

1. Open Visual Web Developer.
2. On the **File** menu, click **NewWeb Site**.

   The **New Web Site** dialog box appears.
3. Under **Visual Studio installed templates**, click **ASP.NET Web Site**.
4. In the **Location** box, enter the name of the folder where you want to keep the pages of your Web site.

   For example, type the folder name **C:\WebSites**.
5. In the **Language** list, click the programming language you prefer to work in.
6. Click **OK**.

   Visual Web Developer creates the folder and a new page named Default.aspx.

Creating a Theme

A theme is a collection of property settings that allow you to define the look of pages and controls. You can apply this look consistently across pages in a Web application. Themes are made up of several elements: server control skins, CSS files, and other resources. In this example, you use a skin and a style sheet to create a theme.

Themes are defined in special directories in your Web site project.

To create a theme

1. In Solution Explorer, right-click the Web site project name, click **Add ASP.NET Folder**, and then click **Theme**.

   The App_Themes folder is created automatically and a new themes folder named Theme1 is added.
2. Right-click the new Theme1 folder, and click **Rename**. Type **Blue** and press ENTER.

3. Right-click the new **Blue** folder, and then click **Add New Item**.

4. In the **Add New Item** dialog box, select **Skin File** and name the file **default.skin**. Click **Add**.

5. In Solution Explorer, right-click the new **Blue** folder again, and then click **Add New Item**.

6. In the **Add New Item** dialog box, select **Style Sheet**. Name the style sheet **default.css**. Click **Add**.

   The first theme is now created with an empty CSS file and server control skin file. You will edit these files in a moment, but first you need to create a page that contains a control and some HTML that the theme can be applied to.

Creating a Master Page

To show that a theme can easily be applied to both a master page and a page that uses that master page, create a simple master page to use with the Default.aspx page in your Web project.

To create the master page

1. In Solution Explorer, right-click the name of your Web site, and then click **AddNew Item**.

2. Under **Visual Studio installed templates**, click **Master Page**.

3. In the **Name** box, type **Master1.master**.

4. In the **Language** list, click the programming language you prefer.

5. Clear the **Place code in separate file** check box, and then click **Add**.

   The new master page opens in Source view. At the top of the page is an @ Master declaration instead of the @ Page declaration normally found at the top of ASP.NET pages. The body of the page contains an **asp:contentplaceholder** control with the ID property set to `ContentPlaceHolder1`, which defines the area of the master page where replaceable content will be merged from content pages at run time. You will work more with the content placeholder later in this walkthrough.

Laying Out the Master Page

The master page defines the elements that make up the page. It can contain any combination of static text and controls. A master page also contains one or more content placeholders that designate where dynamic content will appear when pages are displayed.

In this walkthrough, you use a table containing a title, several horizontal rules, and a master page content placeholder as the layout for your Home.aspx page.

To create a table for the master page

- With the **Master1.master** file selected in Source view, select the text between the two **form** elements and paste the following content into the selected area. Note that this code puts the content placeholder in a table, instead of between the **div** elements as it is in the default master layout.

  C#

  ```
          <table width="100%" cellspacing="0" cellpadding="0" border="0" class="header">
              <tr>
                  <td class="title">Switchable Themes Example</td>
              </tr>
              <tr>
                  <td><hr /></td>
              </tr>
              <tr>
                  <td>
                  <asp:contentplaceholder id="ContentPlaceHolder1" runat="server">
                  </asp:contentplaceholder>
                  </td>
              </tr>
              <tr>
                  <td><hr /></td>
              </tr>
          </table>
  ```

  The master page now has a layout that can be applied to the content page (named Home.aspx) that you will create in the next section.

Creating a Content Page

The master page provides the template for your content. You define content for the master page by creating an ASP.NET page that is associated with the master page. The content page is a specialized ASP.NET page that contains only the content to be merged with the master page. In the content page, you add the text and controls that you want to display when users request that page.

The content page will use the master page you have created and the themes you have yet to finish. The page will use the master page's content placeholder and have a title, a subtitle, and a drop-down list. Because the page will be using a master page, it must contain a **MasterPageFile** attribute in the @ Page directive, as well as the content placeholder.

To create the content page

1. In Solution Explorer, right-click the name of your Web site, and click **Add New Item**.
2. Under **Visual Studio installed templates**, click **Web Form**.
3. In the **Name** box, type **Home**.
4. In the **Language** list, click the programming language you prefer.
5. Select the **Select master page** check box, and then click **Add**.

   The **Select a Master Page** dialog box appears.

6. Click **Master1.master**, and then click **OK**.

   A new file called Home.aspx is created. The page contains an @ Page directive that attaches the current page to the selected master page with the **MasterPageFile** attribute, as shown in the following example.

   C#

   ```
   <%@ Page Language="C#" MasterPageFile="~/Master1.master" ... %>
   ```

   The page also contains an **<asp:Content>** element that you will work with next.

Adding Content to the Content Page

A content page does not have the usual elements that make up an ASP.NET page, such as **html**, **body**, or **form** elements. Instead, you add only the content that you want to display on the master page by replacing the placeholder regions you created in the master page. For this example, you can add a heading 1 element, a heading 2 element, a paragraph element, and a drop-down list. You will use the drop-down list to select a theme to apply to the page.

To add content to the home page

1. Paste the following code between the **asp:Content** beginning and ending tags for the placeholder with the ID property set to `ContentPlaceHolder1`. The following example creates three HTML elements: a heading 1, a heading 2, and a paragraph. It also adds a drop-down list control. Note that you could also add the control to the page in Design view if you prefer.

```
<h1 id="title1">Switchable Themes on a Page</h1><br />
<h2 id="title2">Note how the master page content and the page content are affected</h2>
<p>Select a color from the drop-down list below to change the appearance of this page.</p>
<br /><br />
<asp:dropdownlist id="ddlThemes" runat="server" autopostback="true" >
  <asp:listitem value="Blue">I'd like the page to be blue!</asp:listitem>
  <asp:listitem value="Red">I'd like the page to be red!</asp:listitem>
  <asp:listitem value="Green">I'd like the page to be green!</asp:listitem>
</asp:dropdownlist>
```

2. Add a script section that will run the code that loads the theme when it is selected from the drop-down list. The following example and the **script** tags should be added to the content page on the line after the @ Page directive.

C#

```
<script runat="server">
  public void Page_PreInit()
   {
        // Sets the Theme for the page.
        this.Theme = "Blue";
        if (Request.Form != null && Request.Form.Count > 0)
            this.Theme = this.Request.Form[5].Trim();
    }
</script>
```

The theme is loaded during the **PreInit** event of the page life cycle. The page request form contains an array of values, and the value at index 4 is the value selected from the drop-down list. This value is assigned to the theme of the page, and when the page is loaded, the new theme is applied.

The next step in the walkthrough is to create several themes you can use to show that a different theme is being applied.

Editing the Blue Theme

The Blue theme contains an empty style sheet and an empty skin. Because you know what elements make up the default page and the master page it uses, you can now edit the theme files to add color to the page elements.

To edit the Blue theme

1. Open the Default.skin file from the Blue theme folder in Source view.

2. Add the following code to the drop-down list to designate the page colors when the Blue theme is selected.

```
<asp:dropdownlist runat="server" ForeColor="white" BackColor="Blue" />
```

3. Open the Default.css file from the Blue folder in Source view. First, add the following code to format the table title from the master page.

```
td.title
{
  font-size: 1em;
  text-align: center;
  font-family: verdana;
  font-size: x-large;
  font-weight: bolder;
  color: Navy;
}
```

4. Next, add a background color to the table with the following code.

```
table.header
{
  background-color: Blue;
}
```

5. Next, style the heading 1 and heading 2 elements in the content page.

```
h1
{
  font-size: large;
  color: Navy;
}

h2
{
  font-family: Verdana;
  font-size: medium;
  margin-top: 30;
  color: Navy;
}
```

6. Finally, style the horizontal rule and the paragraph element.

```
p
{
  font-family: Verdana;
  font-size: small;
  color: Aqua;
  text-align: left;
}

hr
{
  border: 0;
  border-top: 2px solid Aqua;
  height: 2px;
}
```

Connecting the Themes to the Page

Before you can see the theme applied to the Home.aspx page, you need to add an attribute to the @ Page directive that indicates that the page uses a theme.

To connect a page to a theme

1. Open Home.aspx in Source view.
2. Add the **StylesheetTheme** attribute to the @ Page directive and set it to equal the Blue theme. The page directive should look like the following example.

C#

```
<%@ Page Language="C#" MasterPageFile="~/Master1.master" Title="Switchable Themes" StylesheetTheme="Blue" %>
```

Testing the Page

You can test the page by running it as you would any ASP.NET page.

To test the page

- While viewing the Home.aspx page, press CTRL+F5 to run the page.

  ASP.NET merges the content in the Home.aspx page and the layout in the Master1.master page into a single page, and then applies the Blue theme and displays the resulting page in the browser. Note that the Blue theme has been applied to the HTML elements and the drop-down list as well as the title and background defined in the master file.

Creating Additional Themes

The Blue theme is fine, but the purpose of this walkthrough is to give the page user several options for a theme. You can copy the skin and style sheet files into new theme directories, and then edit the colors used in the theme to reflect the new theme colors. The following procedure creates two new themes, called Red and Green.

To create additional themes

1. In Solution Explorer, right-click the **App_Themes** folder, click **Add ASP.NET Folder**, and then click **Theme**. With the folder title **Theme1** selected, type **Red** and press ENTER.
2. Open the **Blue** folder and select the Default.skin and Default.css files. Right-click the two selected files, and then click **Copy**.
3. Right-click the **Red** folder, and then click **Paste**.

4. Repeat step 1, but name the new theme folder **Green**. Then, right-click the **Green** folder and click **Paste**, placing copies of the Default.skin and Default.css files into the **Green** folder.

5. Edit the color attribute in the skin file to reflect the theme color. For example, the skin file in the Red theme should look like the following example.

```
<asp:dropdownlist runat="server" ForeColor="white" BackColor="Red" />
```

6. Edit the style sheet for each theme to reflect the theme's name. Note that you will want to use several shades of green for the various HTML elements and text to show up against the background. The style sheet for the Green theme might look like the following example.

```
p
{
  font-family: Verdana;
  font-size: small;
  color: Teal;
  text-align: left;
}

hr
{
  border: 0;
  border-top: 2px solid Teal;
  height: 2px;
}

h1
{
  font-size: large;
  color: Green;
}

h2
{
  font-family: Verdana;
  font-size: medium;
  margin-top: 30;
  color: Green;
}
```

```
table.header
{
  background-color: Lime;
}

td.title
{
  font-size: 1em;
  text-align: center;
  font-family: verdana;
  font-size: x-large;
  font-weight: bolder;
  color: Teal;
}
```

Testing the Theme Selection

The drop-down list can now be used to select among the three themes for the page.

To select different themes

1. Switch to the Home.aspx page, and then press CTRL+F5.
2. Select either **Green** or **Red** from the drop-down list.

   Note that the style sheet is applied to the HTML elements of the page and the skin is applied to the drop-down list control.

Next Steps

For more information about using master pages, see Walkthrough: Creating and Using ASP.NET Master Pages in Visual Web Developer.

You can use a theme property in a user profile to make a user's theme selection persist after the user leaves your site or closes the browser. For information about how to set up user profiles, see Walkthrough: Maintaining Web Site User Information with Profile Properties.

See Also

Tasks

How to: Apply ASP.NET Themes

Other Resources

ASP.NET Master Pages